CIS 215: Homework 5

Deadline: 4/5/2025, by 11:59 PM EST

Deadline for resubmission (after feedback): 4/12/2025, by 11:59 EST

Any assignment turned in blank, close to blank, or after 4/5/2025 **will be graded with penalty**. If you submit multiple attempts before the 4/5/2025 deadline, only the latest one will be graded. For details on the late policy, please read the syllabus.

# Homework 5: Practicing JavaScript

In this homework, you will practice programming in JavaScript. While it is not required, it’s recommended that you continue practicing with Git/GitHub in all assignments.

You may notice this assignment looks familiar, and that’s because it’s the same as homework 3! Now you just need to write the equivalent code for JavaScript. It is up to you whether you complete the same set of tasks or a different set.

The documentation is always required, so make sure you submit complete and accurate documentation with your code. I recommend writing documentation alongside your code.

There are also some **Challenge Problems**, these are for those whose goal is to learn how to code. They contain some harder exercises, where you are not told what functions/loops/etc to create, so you need to decide for yourself. It is good practice for the project, so I recommend attempting at least one of them.

I would recommend completing either the first few (Loops, Arrays, Functions) and one of the lower difficulty challenge problems (FizzBuzz, Anagrams, Palindrome) or some combination of challenge problems that earns at least 10 points and Functions. Commenting Etiquette is also a nice way to get a few extra points so you avoid corrections for minor mistakes.

## 

## Loops

Pick at least 2 tasks from this list:

* Create an array of the even numbers from one to n where n is a variable
* Echo all numbers from 1 to 30 that are divisible by 2 or 3 (or both!)
* Create an indexed array with at least 3 names (do not use a loop to do this). Echo names that are longer than 5 letters long. You should have at least one name less than 5 letters and at least one greater than 5 letters.
* Create an array with at least 5 numbers (do not use a loop to do this). Reverse the array (do not use an existing list function).
* Check that a string has a letter, a number, and a special character (not a letter or number).

Use a for loop and a while loop for each task. Write in a comment which loop you think was the better choice for the task.

## Arrays

Pick an array function we did not discuss in class. Explain, in your own words, in a comment what it does, then write some code demonstrating what it does.

Do not just copy the code from the docs, come up with something different. You can think of this as your own example that you would add to the docs to explain how this function works.

## Functions

Put all the other pieces of code you’ve written into well-named functions. Remember, each function should have a purpose, and if it’s getting long, it may be time to use another function.

In addition, write a function that completes one of the following tasks:

* Any of the loop tasks you have not already completed.
* Given a string, recreate the string with no vowels.
* Reverse a string (do not use an existing string function).
* Given an array and a value, remove all entries of that value from the array (do not use an existing array function to do all the work).
* Given a number, determine the letter grade earned (ignore plus/minus values).

## Scope

Using comments, describe the scope of at least 3 of the variables you created in this assignment and 1 of the functions. You should choose variables such that not all of them have the same scope.

## Commenting Etiquette

At the start of the file, create a multi-line comment describing the functionality of the code in your file. Include your name and school email as an “author.”

Write a multi-line comment over each of your functions describing:

* What it does (briefly)
* The parameters and their types
* The return value(s) and their type(s)

Anywhere you believe your code may be unclear, add an inline comment. You need to do this **and** your documentation, but you are allowed to copy-paste text between the two.

## 

## Challenge Problem: FizzBuzz

There’s a children’s game known as FizzBuzz that is a common interview problem for computer science positions.

The premise is simple: [Count out numbers starting at 1, but say Fizz instead of any multiple of 3, Buzz instead of any multiple of 5, and FizzBuzz for multiples of 3 and 5](https://en.wikipedia.org/wiki/Fizz_buzz).

Write a program which will output the correct sequence of numbers, Fizzes, Buzzes, and FizzBuzzes, from 1 to any given number n.

***Hint***: The [mod operator](https://www.php.net/manual/en/language.operators.arithmetic.php) is very helpful here.

As an additional challenge: try to make your code as adaptable as possible. What if I also wanted you to say “Mazz” at any multiples of 7? What if I wanted you to say “Bozz” instead of “Buzz”? How easy would your code be to adapt to these?

In general, you’re going to need to make some changes, but ideally you should only need to change a few lines of code.

## 

## Challenge Problem: Anagrams

An [Anagram is a rearrangement of letters in a word or phrase to form another word or phrase](https://en.wikipedia.org/wiki/Anagram).

Write code to determine if two inputs are anagrams of each other. The number of spaces in the entries are not important, just that all the letters can be rearranged.

You can test this using the anagrams on the wikipedia page given.

## Challenge Problem: Palindromes

A Palindrome is a word or phrase that is the same forwards and backwards. A simple example is tacocat. A more complex sentence would be “Do geese see God?” You can see more examples on [this Wikipedia page](https://en.wikipedia.org/wiki/List_of_English_palindromic_phrases).

Write code that tests if a string is a palindrome. Remember that capitalization, whitespace, and punctuation do not affect whether a phrase is a palindrome.

You can test this using the palindromes on the wikipedia page given.

# 

# Challenge: Binary Search

Binary Search is an algorithm that allows us to search a sorted list. It is well known for being fast and efficient.

How it works: We have a value, x, that we are checking to see if it is in the list. We check the middle value. If x is equal to the middle value, we’ve found our item! If x is less than the middle value, then we throw out all the elements x and greater and search on the left side of the original list. If x is greater than the middle value, we throw out the left (lesser) side of the list and search on the right side of the original list. This process continues until the value is found or we determine that the value is not in the list.

Here is an example of the process:

We are looking for 2 in [1,2,3,4,5,6,7]

The middle value is 4, and 2 is less than 4, so we restrict our search to [1,2,3]

The middle value is 2, which is equal to the value we want! So the item is in the list.

Another example:

A user on a social media site says "Still laughing about the time a computer scientist who had his bike stolen tried to explain binary search to a cop" with an image of an excerpt from an article that reads "Afterwards I found a chatroom thread among Cambridge computer scientists, one of whom had also been told that unless he could pin down the moment of theft no one would look at the footage. He said he had tried to explain sorting algorithms to police - he was a computer scientist, after all.
You don't watch the whole thing, he said. You use a binary search. You fast forward to halfway, see if the bike is there, and if it is, zoom to three quarters of the way through. But if it wasn't there at the halfway mark, you rewind to a quarter of the way through. It's very quick. In fact, he had pointed out, if the CCTV footage stretched back to the dawn of humanity, it would probably have only taken an hour to find the moment of theft. This argument didn't go down well."

# Documentation

In this class, every homework assignment also requires a written document accompanying the code. Usually, this document will discuss how the code does or does not work. This is practice for real-life coding positions, where you will be expected to explain your own code and identify any potential issues.  
Using the [Documentation Template](https://docs.google.com/document/d/1nofkOmITGh_aTZ8nhNKHm0gzGJ4gNzVj2XOUPwLmhkE/edit?usp=sharing) provided, please complete documentation for your code. For this assignment, you can just use the “complex sections of code” table, as we have not yet covered functions.  
Documentation will be graded on content (does it contain information on everything in the assignment?), technical language (does it use terms appropriately?), and readability (is the documentation easy to read?).

Incorrect or incomplete documentation is grounds for a zero for the entire assignment. This is particularly focused on the content portion. If a significant number of items are missing or the explanation does not align with the code, the student may receive a zero for documentation or a zero for the entire assignment. I highly recommend working on your documentation as you work on your code.

# How Your Grade is Determined

For this homework, you are graded out of 20 points. The rubric section details different ways you may score these points. Typically, you will be able to pick and choose parts of assignments to complete.

Scoring more than 20 points will not give you a grade over 100%. For example, if a student earns 23 points on the assignment, the grade they will get in Brightspace will be 20/20, 100%. Students may go for as many points as they wish, attempting to earn more than 20 is a good strategy to make sure the final grade is as high as possible.

As in any class, plagiarism is strictly prohibited.

## Rubric

A correctly implemented, complete item will score the number designated in the Total Points column. If an item is used incorrectly or is incomplete, it will score a fraction of the total points. For example, If you do not pick two tasks or do not do both loops for each task, you may get a 2/4 for loops.

If you’re stuck on an item, most items have links to the relevant docs or a helpful website. You will be able to reference docs on your exam, so get comfortable with them!

| **Item** | **Total Points** |
| --- | --- |
| Loops | 4 |
| Arrays | 2 |
| Function | 4 |
| Scope | 2 |
| Commenting Etiquette | 4 |
| Challenge Problem: FizzBuzz | 6 |
| Challenge Problem: Anagrams | 8 |
| Challenge Problem: Palindromes | 8 |
| Challenge Problem: Binary Search | 10 |
| Documentation | 6 |

**Assignments not uploaded to CSNLinux will not be graded. Assignments missing documentation will not be graded. Incomplete or incorrect documentation may be grounds for a zero.**

# Submitting this assignment

To submit this assignment, upload your homework5.js file to Brightspace along with your documentation. The documentation should be either .docx (Word) or .pdf.

Also put in the comments a link to your assignment on CSNLinux.